Methodological Aspects in the Design of a Multi-Agent System

Luca Cernuzzi
Universidad Católica "Ntra. Sra. de la Asunción"
Campus Universitario,
Asunción-Paraguay
595-21-334650
lcernuzz@uca.edu.py

Adriana Giret Boggino
Universidad Católica "Ntra. Sra. de la Asunción"
Campus Universitario,
Asunción-Paraguay
595-21-334650
agiret@uca.edu.py

ABSTRACT
Agent design is a complex human activity involving active and proactive modules of software that may present knowledge representation and learning aspects. In this study we review some Agent-Oriented Methodologies including extensions of object oriented and knowledge engineering methodologies. Moreover, we present a case study of a multi-agent system for requirement engineering formal specification that was designed using “the Agent Modelling Technique for Systems of BDI agent” [12]. Finally, a discussion about the use of the methodology is presented.

Keywords
Agent oriented methodologies, software engineering.

1. INTRODUCTION
Agent technology has received a great deal of attention on the last few years. As a result, the industry isbeginning to get interested in using this technology to develop its own products. In spite of the different agent theories, language, architectures and successful agent-based applications developed, very little work for applying (and evaluating) techniques to develop applications using agent technology has been done [12, 13, 14].

Each proposed solution, dealing with design, architectural implementation or methodological problems, inevitably include implicitly or explicitly - a definition of agent. Unfortunately, there is no standard definition; moreover, specialized literature [16, 5] presents a wide range of very different definitions. We adopt the definition proposed by Wooldridge [20], because we believe it includes all the relevant aspects of agenthood.

There are two views of agents, weak and strong:
A weak definition of agency is that an agent is i) autonomous - agents operate without direct intervention and are able to migrate to different processors or networks, ii) social - agents interact with other agents, iii) proactive - agents perceive their environment and respond to changes that occur and iv) pro-active - agents affect their environment rather than passively allowing their environment to affect them.

A strong definition of agency is that an agent, in addition to the above, has one or more of the following: v) mentalistic notions - an agent has beliefs (knowledge), desires, and intentions (pro-active behavior), vi) rationality - an agent performs actions which further its goals, vii) veracity, and viii) adaptability or learning.

In this paper we focus on the use of Agent-Oriented Methodologies in the construction of a multi-agent system. Therefore, in section 2 we introduce a background about different methodologies and approaches offered by specialized literature. In section 3, we present a case study and open a discussion about the use of the Agent Modelling Technique for Systems of BDI agents [12]. Finally, some conclusions are drawn in section 4.

2. AGENT ORIENTED METHODOLOGIES
In Software Engineering context [6] the purpose of a methodology is to promote a certain approach to solving a problem by preselecting the methods and techniques to be used. Having this in mind, we can say that the role of agent-oriented methodologies is to assist an agent-based application in all of its life cycle phases, including its management.

To avoid building a methodology from scratch, the researchers on agent-oriented methodologies have followed the approach of extending existing methodologies to include the relevant aspects of the agents [8]. These extensions have been carried out mainly in two areas: object oriented (OO) methodologies and knowledge engineering (KE) methodologies.

2.1 Extensions of Object Oriented Methodologies
Several reasons justify the approach of extending object-oriented methodologies to cope with the design of agent systems. First of all, there exists some similarities between the object-oriented and the agent concepts [5, 27]. Second, the commonly usage of object-oriented languages to implement agent-based systems. Furthermore, advantages and maturity offered by object-oriented methodology may be a suitable starting point in the construction process of agent based systems.

In this section the following agent-oriented methodologies are reviewed: Agent Oriented Analysis and Design [1], Agent Modelling Technique for Systems of BDI agents [12], MASB [14, 13] and Agent Oriented Methodology for Enterprise Modelling [11].

2.1.1 Agent Oriented Analysis and Design
This technique [1] defines three models for analysing an agent system: the agent model, that contains the agents and their internal structure (beliefs, plans, goals, ...); the organisational model, that describes the relationships between agents (inheritance and roles in the organisation); and the cooperation model, that describes the interactions between agents.

2.1.2 Agent Modelling Technique for Systems of BDI agents
This method [12] defines two main levels (external and internal) for modelling BDI (Belief, Desire and Intention) agents.
The external viewpoint consists of the decomposition of the system into agents and the definition of their interactions. This is carried out through two models: the agent model, for describing the hierarchical relationship between agents and the relationships between concrete agents; and the interaction model, for describing the responsibilities, services and interaction between agents and external systems.

The internal viewpoint carries out the modelling of each BDI agent class through three models: the belief model, which describes the beliefs about the environment; the goal model, which describes the goals and events an agent can adopt or respond to; and the plan model, which describes the plans an agent can use to achieve its goal.

2.1.3 Multi-Agent Scenario-Based Method (MASB method)

This method [14,13] is intended to be applied for multi-agent systems (MAS) in the field of cooperative work. The analysis phase consists of the following activities: Scenario description, Role functional description, Data and world concepts, and System-user interaction modelling. The design phase consists of: MAS architecture and scenario description, Object modelling, Agent modelling, Conversation modelling and System design overall validation.

2.1.4 Agent oriented methodology for Enterprise modelling

This methodology [11] proposes the combination of object-oriented methodologies (OOSE [21]) and enterprise modelling methodologies IDEF (Integration Definition for Function modelling) [4] and CIMOSA (Computer Integrated Manufacturing Open System Architecture) [10]. The identified models are:

- Functional Model: describes the functions (inputs, outputs, mechanism and control) using IDEF0 diagrams.
- Use Case Model: describes the actors involved in each function.
- Dynamic Model: analyses object interactions.
- The Agent Oriented System: includes a compound of Agent Identification, Coordination protocols and scripts, Plan invocation, and Beliefs, Sensors and Effectors.

2.2 Extensions of Knowledge Engineering Methodologies

Knowledge engineering methodologies can provide a good basis for multi-agent system modelling since they deal with the development of knowledge based systems. Since the agents have cognitive characteristics, these methodologies can provide suitable methods and techniques for modelling the agent aspects related to learning and knowledge representation.

There exist different knowledge engineering methodologies. Among them the CommonKADS [18] is widely considered a European standard. Several solutions [2, 9, 15, 19] have been proposed for multi-agent systems modelling extending CommonKADS [18].

We will review the extensions CoMoMAS [7] and MASCommonKADS [8].

2.2.1 The CoMoMAS methodology

Glaser [7] proposes an extension to the methodology CommonKADS [18] for MAS modelling. The following models are defined:

- Agent Model: defines the agent architecture and the agent knowledge. They are classified as social, cooperative, control, cognitive and reactive knowledge.
- Expertise Model: describes the cognitive and reactive competence of the agent.
- Task Model: describes the task decomposition and details if the task is solved by a user or an agent.
- Cooperation Model: describes the cooperation between the agents, using conflict resolution methods and cooperation knowledge.
- System Model: defines the organisational aspects of the agent society along with the architectural aspects of the agents.
- Design Model: collects the previous models in order to operationalise them, together with the non-functional requirements.

2.2.2 The MAS-CommonKADS methodology

This methodology [8] extends the models defined in CommonKADS, adding techniques from object-oriented methodologies and from protocol engineering to describe the agent protocols. The methodology starts with a conceptualisation phase which is an informal phase used to collect the user's requirements and to obtain a first description of the system from the user's point of view.

The methodology defines the following models:

- Agent Model: describes the main characteristics of the agents, including reasoning capabilities, skills, services, goals, etc.
- Task Model: describes the tasks (goals) carried out by agents, and task decomposition.
- Expertise Model: describes the knowledge needed by the agents to carry out the tasks. The knowledge structures distinguishes domain, task, inference and problem solving knowledge.
- Coordination Model: describes the conversations between agents, that is, their interactions, protocols and required capabilities. The development of the model defines two milestones. The former is intended to identify the conversations and the interactions. The latter is intended to improve these conversations with more flexible protocols such as negotiation and identification of groups and coalitions.
- Organisation Model: describes the organisation in which the MAS is going to be introduced and the organisation of the agent society.
- Communication Model: details the human-software agent interactions, and the human factors for developing these interfaces.
- Design Model: collects the previous models and is subdivided into three submodels, application design, architecture design and platform design.

3. USING METHODOLOGIES IN PRACTICES

In this section we present a case study of a multi-agent system that assists software engineers during formal specification activities. Before describing the agent and its design process, it is important to introduce some methodological considerations.
Among all the above mentioned methodologies, we preselect two of them to assist us during the design and implementation process. Within the extension of object oriented approach, we find that the “Agent Modelling Technique for Systems of BDI agents” (proposed by Kinny et al. [12]) is particularly interesting. In effect, it covers details inherent to mentalistic notions (beliefs, desires and intentions), as well as the roles and responsibilities of each agent within the application domain. Moreover, this methodology covers the specification of necessary interaction between different agents that compose the system.

Within the extension of knowledge engineering approach, we perceive that the “MAS-CommonKADS methodology” (proposed by Iglesias et al. [8]) may be very useful for different reasons. First, the conceptualisation phase has a high level of abstraction; in this way it facilitates the identification of initial components of systems. Second, the methodology deeply covers the specification of the agent intelligence.

In order to make a final decision about the methodology to be used, we need to consider the characteristics of the specific application.

3.1 Agent description

Our case study multi-agent system has been developed for the domain of requirements engineering. Basically the multi-agent system is an agent society that helps to produce formal specification from informal collected data. The society consists of one FORMATTER and one ASSISTANT for each user of the system.

The FORMATTER’s main work is to provide an algebraic specification in two defined formats: functional or non-functional. It also has to monitor user’s repetitive actions that can be automated, and recommend this automation as necessary.

The ASSISTANT structure is a little different. Every time there is only one ASSISTANT (the actual user’s assistant: ASSISTANT_1) that can communicate directly with the actual user. The others’ suggestions are filtered through ASSISTANT_1 that decides about their relevance.

The ASSISTANT_1’s main work is to obtain an internal representation of the user’s specifications and analyze it for possible inconsistencies or ambiguities. It also has to communicate it to the other ASSISTANTS, listen for their recommendations and finally, make suggestions to the user, based on what it thinks is better.

3.2 The Agent Construction Process

To assist the construction process we finally decide to adopt the “Agent Modelling Technique for Systems of BDI agents” proposed by Kinny et al. [12]. Two main reasons support the selection of this methodology: first, it contemplates the adopted agent definition; and second, it covers the characteristics of our multi-agent system that includes some BDI agents.

Like we addressed in Section 2, the “Agent Modelling Technique for Systems of BDI agents” defines two main levels (external and internal) for modelling BDI (Belief, Desire and Intention) agents.

3.2.1 External View: Agent Model

The Agent Model, describes the hierarchical relationships among different abstract and concrete agent classes, and identifies the agent instances which may exist within the system.

Roles of the Application Domain

- ASSISTANT: assists the user by identifying ambiguities and conflicts, recommending components or sentences about the specification domain.
- FORMATTER: formats the requirements document.
- ADMINISTRATOR: administers the data repository.

Agent Class Model

An Agent Class Model is a set of class diagrams, which defines abstract and concrete agent classes and captures the inheritance and aggregation relationships between them. Each diagram is a directed, acyclic graph containing nodes denoting abstract classes (A) and concrete agent classes. Edges with a triangle with a vertex denote inheritance, and edges with a diamond denote aggregation. Figure 1, presents the agent class diagram of the FORMATTER, ASSISTANT and ADMINISTRATOR agents. FORMATTER can be specialised in a Functional FORMATTER or in a Non-Functional FORMATTER, while ASSISTANT is an aggregation of a Recommender and an Analyzer.

![Agent Class Diagram](image1)

Agent Instance Model

An agent instance model is an instance diagram which defines: the static agent set (S), which are instantiated at compile time, and the dynamic agent set, which may be instantiated at run time. A multiplicity notation at the instance and of the instantiation link may be used to indicate whether a dynamic class may be multiply instantiated. In Figure 2, the FORMATTER and the ADMINISTRATOR are static agents that can have only one instance, while the ASSISTANT is dynamic and may have multiple instances.

![Instance Diagram](image2)
3.2.2 External View: Interaction Model
The Interaction Model describes the responsibilities of an agent class, the services it provides, associated interactions, and control relationships between agent classes.
Modeling of agent interactions is developed in KQML [3].
For space reasons, at this point we only present a reduced Interaction Model for the ASSISTANT.
ASSISTANT is an aggregation of a recommender and an analyser (when the user starts the application all the ASSISTANTS wake up and the user’s ASSISTANT becomes ASSISTANT_1).

Responsibility: recommend to other ASSISTANTS.
Services: Monitor for compiled sentences.
KQML message broadcast
:from ASSISTANT_1
:content compiled sentences
:ontology requirement
:language ASCII
:sender ASSISTANT_1
:receiver recommender

3.2.3 Internal View: Belief Model
The Belief Model describes the information about the environment and internal state that an agent of that class may hold, and the actions it may perform.
The notation used in the diagrams is an OMT [17] like notation.

3.2.4 Internal View: Goal Model
The Goal Model describes the goals that an agent may possibly adopt, and the events to which it can respond.
Table 1 specifies the ASSISTANT’s goals. Only three types of goals can be specified by the model: achieve(!), verify($) and test(?).

<table>
<thead>
<tr>
<th>ASSISTANT</th>
<th>Recommend(Assistant, User) $</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Communicate(String) !</td>
</tr>
<tr>
<td></td>
<td>exist_equal(String, Doc. Req) ?</td>
</tr>
<tr>
<td></td>
<td>-conflict(String) !</td>
</tr>
<tr>
<td></td>
<td>-ambiguity(String) !</td>
</tr>
<tr>
<td></td>
<td>compiled_sentence(String) !</td>
</tr>
<tr>
<td></td>
<td>new_sentence(Doc. Req) !</td>
</tr>
<tr>
<td></td>
<td>analyzer(Doc Req) !</td>
</tr>
<tr>
<td></td>
<td>communicate(String) !</td>
</tr>
</tbody>
</table>

| Table 1: ASSISTANT’s goals |

3.2.5 Internal View: Plan Model
The Plan Model describes the plan that an agent may possibly employ to achieve its goals. A plan model consists of a set of plans. Individual plans are specified as plan diagrams, which are denoted by a form of class icon. The plan graph is a state transition diagram, similar to an object-oriented dynamic model.
For space reasons, in Figure 4 we only present one plan model for the ASSISTANT.
3.3 Discussion

In the former sections, we briefly presented the design of the system. In this subsection we introduce some considerations about the design process using Agent Modelling Technique for Systems of BDI agents.

In our experience, we consider the methodology to be very helpful in ordering and structuring the design activities. We particularly point out some very interesting aspects:

- guidance offered in identifying which entity may be considered an agent;
- clear separation of external view (capturing architectural aspects of the global system) and internal view (centered on specific agent design);
- support for communication between agents (specification of messages order and parameters);
- special focus on the roles and responsibilities of each agent in the system;
- explicit specification of the mentalistic notions of agents.

However, some constraints are imposed by the Agent Modelling Technique for Systems of BDI agents.

As seen in subsection 3.1, the methodology is based on two views of the system: an internal and an external view.

In identifying the agents of the system (external view) the methodology allowed us to identify the ADMINISTRATOR agent, but this agent conflicts with the definition that supports the methodology. In effect, it is impossible to find belief, desire and intention components for this entity.

In the internal view, we developed three models: beliefs model, goals model and plans model.

In the Beliefs Model we specified the agent’s knowledge about its environment. This included class diagrams and the specification of derived functions and first order logic predicates about the elements of the agent’s knowledge. First order logic theory is a very powerful formalism. However, since it requires a binary result, it does not lie to specify uncertainty requirements that may be involved or necessary in some environments, where the agent can only partially rely on its environment knowledge. Another limitation is that the methodology doesn’t address the possibility to add, delete or modify beliefs of the belief set at run time. This functionality may be very important for modelling agents’ learnings.

Furthermore, in modelling the agent’s goals, the methodology is limited to only three types of goals: achieve, verify and test. In our experience, this limitation is quite reductive because sometimes it is necessary to specify fuzzy goals. Moreover, the specification of an evolving behavior of agents is not supported.

Finally, we consider that the Agent Modelling Technique for Systems of BDI agents does not consistently cover all the life cycle stages of agent construction process. In effect, agent systems designers may profit from the methods and techniques proposed on the first steps of design processes, while lacking of the adequate orientation in translating from the design to the implementation. Consequently, all results obtained during design may be frustrated because they do not meet a consistent and straightforward way to incrementally derive in a good modular code. Probably, a CASE tool may reduce the mentioned gap.

4. CONCLUDING REMARKS

Agent design is a complex human activity involving active and proactive modules of software that may include knowledge representation and learning aspects.

Specific design techniques, models, and methodologies may provide a very important conceptual support for agent based systems construction. In this study we reviewed some Agent-Oriented Methodologies including extensions of object oriented [1, 11, 12, 13, 14] and knowledge engineering [7, 8] methodologies.

Considering the different proposals, designers have to select (or eventually create) a specific methodology to be used in the construction process.

In this paper we present a case study of a multi-agent system using "the Agent Modelling Technique for Systems of BDI agent" [12]. The multi-agent system, concerning the domain of requirement engineering, is basically an agent society that helps software engineers to elaborate formal specification.

In section 3.3 we present some considerations about the experience. We found that the methodology is helpful in structuring the designing activities focusing on agent’s roles, responsibility and mentalistic notions as well as in architectural aspects. However, we found some limitations. Mainly, the impossibility to specify uncertainty and evolutionary behavior of agents.

There are issues remaining for future work. Probably, one of the most interesting one is to make a comparison of different methodologies, in different environments and application domains.

5. REFERENCES


